7-2 Project Two

LaNise Essick

CS-320 Software Test Automation& QA

The importance of unit testing in the development of software solutions cannot be overstated. By finding and addressing defects quickly, it reduces the risk of running into program errors. In order to align with the software requirements. Software requirements were laid out for each service and class at the beginning of each module. For example, the contact class had the following requirements:

* The contact object shall have a required unique contact ID string that cannot be longer than 10 characters. The contact ID shall not be null and shall not be updatable.
* The contact object shall have a required firstName String field that cannot be longer than 10 characters. The firstName field shall not be null.
* The contact object shall have a required lastName String field that cannot be longer than 10 characters. The lastName field shall not be null.
* The contact object shall have a required phone String field that must be exactly 10 digits. The phone field shall not be null.
* The contact object shall have a required address field that must be no longer than 30 characters. The address field shall not be null.

The methods I wrote were parameterized, and unit tests were used to confirm the functionality of each method. I have provided a screen shot of method written for the contact class (IMG.1).

(IMG.1)

Text

Description automatically generated

My coverage percentage was a %100 for each unit test I performed. I made incorrect data entries to test the functionality of my class and methods. Based on the false data entries my unit test would throw errors set. This is how I was able to confirm each class function worked as it should. To make sure my code was technically sound, I made sure to include notes, and that my code had no errors or warnings. Also, I tested parameters using JUnit test.

Text

Description automatically generated

The code above shows parameters I had set for a test contact, and below is the test requirement used to confirm if a Contact ID entered by a user is two long keeping in mind that the contact ID string cannot be longer than 10 characters and the contact ID could not be null and could not be updatable. To make sure my code was efficient I tested each requirement using Junit testing.

Text

Description automatically generated

The code above illustrates testing I used for my setter methods. By placing these tests in place, executing them, and confirming expected results, I was able to insure the efficiency of my code.

**Reflection**

A software technique I used was Dynamic testing, which tests the dynamic behavior of software code. By using this technique, I was able to check the behavior of the various dynamic variables that are not constant during the runtime of the software. I also identified weak areas. The software technique that I did not apply was static testing which is a way of testing the code without executing it. The major difference between static and dynamic testing is that the identification of defects cannot be easily found by dynamic testing. This is because it involves development standard breaches and the detection of dependencies and inconsistencies in software models.

By changing my mindset to more of a software tester, I was able to ensure that I captured all the test cases. Tests were conducted with caution in various cases. It was essential to appreciate the complexity and interrelationships of the code I was testing. This is because each of the class objects had some requirements that must be fulfilled to certify the code as working correctly. It is necessary for an individual to understand what each section of code is doing to accomplish this. Like areas where each unit of the class had some specific requirement to be able to function. This can be accomplished using a contact service that can perform some crude operations effectively.

I believe that when it comes to reviewing your code, bias can be a major factor that needs to be minimized. Whenever a developer is confident about the code they wrote, they may not test it thoroughly. Since you created variables in a class, you can be confident that the code is effective since you created them. We all should try to limit that when it comes to software testing.

As a software engineer, discipline is key to ensuring quality. The reason is that it will result in a smoother process of creating an application with fewer bugs and errors. When writing or testing code, it is very important not to cut corners. By testing the code properly, money can be saved as well as time and effort. The code is tested before it is shipped to production, so everything is working as it should. For example, using the Contact service ensures each contact has a unique ID that can't be updated. This is done to ensure each contact is unique and can't be interchanged with another contact to avoid errors.